36. The system of claim 32 further comprising means for defining a specific behavior when a function within the source code is called; means for storing the defined information; means for compiling the defined information as a separate object; and means for linking the compiled object to the code.

37. (Amended) A computer readable medium having stored thereon a set of instructions including instruction for testing a computer program, the instructions, when executed by a computer, cause the computer to perform the steps of:

A8

parsing a source code of the computer program to identify functions in the source code;

generating stubs for the source code responsive to the identified functions;

[illegible]

instrumenting the parsed source code with the generated stubs;

compiling the instrumented code;

testing the compiled code; and

reporting test results.

REMARKS

Claims 1-37 are pending in the application. Claims 1, 2, 7, 10, 11, 14, 19, 27, 32, 33, and 37 are amended. Claims 1-14, 19-27, 32-35, and 37 are rejected under 35 U.S.C. § 102(b) as being fully anticipated by Grossman et al. (U.S. 6,332,213); claims 15, 28 and 36 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Grossman in view of Beizer, "Software Testing Techniques," 1986. Applicants submit that all of the pending claims, as amended, are patentable over the cited references, and reconsideration and allowance of these claims are respectfully requested.

The amended independent claims 1, 32, and 37 include, among other limitations, "parsing a source code of the computer program to identify functions in the source code," and "responsive to the identified functions, generating stubs for the source code." Independent claim 19 includes, among other limitations, "parsing the source code of the computer program to identify a plurality of smaller components in the source code;" and "based on the identified plurality of smaller components, generating stubs to replace some of the identified plurality of smaller components."

However, Grossman does not teach or suggest "parsing a source code of the computer program to identify functions in the source code," as required by the independent claims 1, 32, and 37. Similarly, the method of Grossman does not perform "parsing the source code of the computer program to identify a plurality of smaller components in the source code," as required by the independent claim 19. Instead, Grossman describes parsing (and instrumenting) an <u>intermediate representation</u> of the source code.

For example, at the beginning of the specification, Grossman emphasizes a method for instrumenting a computer program by examining an initial <u>intermediate representation</u> (IR) of the program, selecting <u>portions of the initial intermediate representation for instrumentation</u>, and instrumenting the portions. (Col. 2, lines 62-66, emphasis added. In Grossman, "[t]he parse tree data element 63 is provided to the third stage 54 of the compiler 42 which uses the data from the parse tree data element 63 to produce Intermediate Representation (IR) data that is stored in an IR data element 64. As described in more detail hereinafter, the IR data element 64 contains an intermediate representation of the program that is independent of the particular language used for the source code 44 and is also independent of the target processor on which the object code 46 will execute." (Col. 6, lines 6-14, and FIGs. 3-4.)

Furthermore, Grossman does not teach or suggest "responsive to the identified functions, generating stubs for the source code" as required by the independent claims 1, 32, and 37. The section in

Grossman (col. 17, lines 34-50), cited in the Office action (page 3, first paragraph) does not teach or suggest the above limitation. By requiring an "executable library," Grossman is clear about using a (already created) library of functions (stubs), instead of "responsive to the identified functions, generating stubs for the source code." For example, in col. 17, lines 40-45, Grossman describes that "[t]he initialization routine determines if an executable library corresponding to the run time instrumentation routine is available. If not, then the addresses of the functions that are called indirectly by the indirect function calls added by instrumentation are set to 'stub' routines that simply return without executing anything." Also, "[i]f, on the other hand, the initialization routine determines that the executable library for providing instrumentation during run time is available, then the addresses of the functions that are called indirectly by the instrumentation nodes are set to the instrumentation routines." Col. 17, lines 51-55.

For the same reasons mentioned above, Grossman does not teach or suggest the limitation of "based on the identified plurality of smaller components, generating stubs to replace some of the identified plurality of smaller components," as required by the independent claim 19. As a result, independent claims 1, 19, 32, and 37 are not anticipated by Grossman.

Dependent claims 2 and 33, include, among other limitations, "generating source code for replacing the name of externally called functions within the source code with the name of specific functions with same signature as the externally called functions." This limitation is <u>not</u> "inherent from the teaching of 'run time instrument code by using a separate routine', where the separated routine is an external routine in a library (column 17, lines 33-50)." See Office action, page 3, second paragraph. In deed, the routine in a library is different from "generating source code for replacing the name of externally called functions within the source code." Thus, dependent claims 2 and 33 are not anticipated by Grossman either.

As a result, neither Grossman, nor Beizer, alone or in combination, teach or suggest the above-mentioned limitations, required by independent claims 1, 19, 32 and 37, and dependent claims 2 and 33. The remaining dependent claims 2-18, 20-31 and 34-36 all depend, directly or indirectly from their respective independent claims. Therefore, these claims are also patentable over the cited references, as being dependent from allowable independent claims and for the additional limitations they include therein.

In view of the foregoing amendments and remarks, it is respectfully submitted that this application is now in condition for allowance, and accordingly, reconsideration and allowance are respectfully requested.

Attached hereto is a marked-up version of the changes made to the claims by the current amendment. The attached page is captioned "**Version with markings to show changes made**."

Respectfully submitted,

CHRISTIE, PARKER & HALE, LLP

By [illegible]
Raymond R. Tabandeh
Reg. No. 43,945
626/795-9900

RRT/clv

VERSION WITH MARKINGS TO SHOW CHANGES MADE

Claims 1, 2, 7, 10, 11, 14, 19, 27, 32, 33, and 37 are amended as follows:

1. (Amended) A method for testing a computer program comprising the steps of:
parsing a source code of the computer program to identify functions in the source code;
~~creating~~ responsive to the identified functions, generating stubs for the source code;
instrumenting the parsed source code with the ~~created~~ generated stubs;
compiling the instrumented code;
testing the compiled code; and
reporting test results ~~in a~~.

2. (Amended) The method of claim 1 wherein, the step of ~~creating~~ generating stubs comprises generating source code for replacing the name of externally called functions within the source code with the name of specific functions with same signature as the externally called functions.

7. (Amended) The method of claim 6 wherein, the user-specified functions are specified within ~~the~~ a GUI.

10. (Amended) The method of claim 1 wherein, the step of ~~creating~~ generating stubs comprises reconstructing a class by removing the source code that is not related to the class.

11. (Amended) The method of claim 1 wherein, the step of ~~creating~~ generating stubs comprises reconstructing a class by ignoring the source code that is not related to the class.

14. (Amended) The method of claim 13 further comprising displaying the monitored test coverage in ~~the~~ a GUI as the test progresses.

19. (Amended) A method for testing a computer program having a source code comprising the steps of:
parsing the source code of the computer program to identify a plurality of smaller components in the source code;
breaking down the source code into ~~a~~ the plurality of smaller components;
based on the identified plurality of smaller components, generating stubs to replace some of the identified plurality of smaller components;

testing the plurality of smaller components individually; and

reporting test results ~~in a GUI~~.

27. (Amended) The method of claim 26 further comprising displaying the monitored test coverage in ~~the~~ a GUI as the test progresses.

32. (Amended) A system for testing a computer program comprising:

means for parsing a source code of the computer program to identify functions in the source code;

means for ~~creating~~ generating stubs for the source code responsive to the identified functions;

means for instrumenting the parsed source code with the ~~created~~ generated stubs;

means for compiling the instrumented code;

means for testing the compiled code; and

means for reporting test results ~~in a GUI~~.

33. (Amended) The system of claim 32 wherein, the means for ~~creating~~ generating stubs comprises ~~means for~~ generating source code for replacing the name of externally called functions within the source code with the name of specific functions with same signature as the externally called functions.

37. (Amended) A computer readable medium having stored thereon a set of instructions including instruction for testing a computer program, the instructions, when executed by a computer, cause the computer to perform the steps of:

parsing a source code of the computer program to identify functions in the source code;

~~creating~~ generating stubs for the source code responsive to the identified functions;

instrumenting the parsed source code with the ~~created~~ generated stubs;

compiling the instrumented code;

testing the compiled code; and

reporting test results ~~in a GUI~~.